Tennis Predictor

(machine learning using python)

**By:-**

**Anshu Kumar, Techno India Salt Lake 13000215025**

**Table of Contents**

Acknowledgement ................................................................................................ 4

Objective ............................................................................................................... 5

Scope .................................................................................................................... 6

Requirement Specification……………….................................................................... 7

Database Design ................................................................................................... 8

Application Work Flow Diagram ........................................................................... 9

Future Enhancement............................................................................................10

Code ……............................................................................................................... 11

**Acknowledgement**

I take this opportunity to express my profound gratitude and deep regards to my faculty*,* for their exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him/her time to time shall carry me a long way in the journey of life on which i am about to embark. I am grateful for their cooperation during the period of my assignment.

*Anshu kumar (13000215025)*

**Project Objective:**

*The primary project goals:*

• To predict the winner of a tennis match

• Split into train and test data and create 4 different types of models from the data - s;lds;ldsDecision Trees, Linear Regression, Native Bayes and K-NN

• Do performance evaluation of each model

**Project Scope**

Best scope of the project can be found in betting:

There are two main categories of tennis betting: pre-game and in-game, with the distinction that pregame bets cannot be placed after the game commences. Furthermore, it is usually possible to bet on a variety of factors, such as the winner of the match, the score of different sets, the total number of games, etc. We will focus on pre-game bets on the winner of the match, as the odds for this bet type are most available historically, allowing us to perform a more comprehensive evaluation of the performance of our model against the betting market

**Requirement Specification**

*(This section contains the business challenge that the application is solving from the point of view of a user of the system)*

• Overfitting :

it is important to note that the performance of players in an upcoming match will need to be estimated based on their past matches. Only recent matches on the same surface against similar opponents accurately reflect the expected performance of the players. For this reason, tennis modelling inherently suffers from a lack of data. The lack of data often results in overfitting of the model, meaning that the model describes random error or noise in the data, instead of the underlying relationship

To overcome the overfitting problem, only the most relevant features of matches will be used for training. The process by which these features are selected is called feature selection, for which various algorithms exist. Removing irrelevant features will also improve training times.

• Functional Requirements :

All data processing components of the system were implemented in the Python1 programming language. Python has several packages for scientific computing which have made the implementation succinct and efficient, in particular NumPy and Pandas. These two libraries provide a clean interface to in-memory manipulation of large

**Database Design**

*(This section describes the Entity Relationship (ER) Diagram and description of the table structure)*

Results for the men's ATP tour date back to January 2000, including Grand Slams, Masters Series, Masters Cup and International Series competitions. Metadata can be found here: <http://www.tennis-data.co.uk/notes.txt>

**Application Work Flow**

*(This section displays the flow of information in the application)*



When evaluating the model, we will only consider predictions for the 20% of the matches, when ordered by uncertainty. we defined uncertainty for a match based on the weights assigned by player’s attributes and surfaces during feature extraction. As we are more confident in the accuracy of features for matches with lower uncertainty, we expect to make a greater profit when betting on these matches. Therefore, we will not place bets on matches with high uncertainties.

***www.globsyn***

***finishingschool***

***.com***

10

**Future Scope of Improvements**

*(This section will list the future aspects of the application which can be incorporated to improve the functionality and user experience)*

Tennis betting expert Peter Webb claims that over 80% of the overall money wagered on tennis matches is bet in-play, i.e., during the course of the match.3 The stochastic models can predict the match outcome probability from any starting score, allowing for in-play betting. Our ML models are not currently capable of adjusting a prediction according to the progression of the match. We could attempt to encode the current score as a match feature, but we doubt that this could compete with the structured hierarchical approaches

We can also consider some more features like gap between two matches of a player, duration of match played to improve the accuracy of prediction.

**Code**

import pandas as pd

import numpy as np

import math

import matplotlib.pyplot as plt

from sklearn import preprocessing, cross\_validation, neighbors,tree

from sklearn.linear\_model import LinearRegression

df =pd.ExcelFile("E:\Worksheet in Session 14 - Project.xlsx")

f=df.parse("atp\_matches\_2000")

f.to\_csv('csvfile.csv',encoding='utf-8',index=False)

hb=pd.read\_csv("csvfile.csv",nrows=1000)

fe=pd.DataFrame()

fe=hb.loc[:,['surface','winner\_name','winner\_rank\_points','winner\_age','winner\_hand','winner\_ht','loser\_name','loser\_rank\_points','loser\_age','loser\_hand','loser\_ht','tourney\_level']]

fe.fillna(value=-99999, inplace=True)

player\_names=pd.unique(fe[['winner\_name', 'loser\_name']].values.ravel('K'))

a=pd.factorize(player\_names)

p\_uid=dict(zip(a[1],a[0]))

sur\_name=pd.unique(fe[['surface']].values.ravel('K'))

p\_hand=pd.unique(fe[['surface']].values.ravel('K'))

b=pd.factorize(sur\_name)

s\_uid=dict(zip(b[1],b[0]))

fe["winner\_id"]=fe["winner\_name"]

fe["winner\_id"].replace(p\_uid,inplace=True)

fe["loser\_id"]=fe["loser\_name"]

fe["loser\_id"].replace(p\_uid,inplace=True)

fe["surface\_id"]=fe["surface"]

fe["surface\_id"].replace(s\_uid,inplace=True)

fe["winner"]=fe["winner\_id"]

w\_hand=pd.unique(fe[['winner\_hand']].values.ravel('K'))

w\_hd=pd.factorize(w\_hand)

w\_huid=dict(zip(w\_hd[1],w\_hd[0]))

fe["winner\_hd"]=fe["winner\_hand"]

fe["winner\_hd"].replace(w\_huid,inplace=True)

l\_hand=pd.unique(fe[['loser\_hand']].values.ravel('K'))

l\_hd=pd.factorize(l\_hand)

l\_huid=dict(zip(w\_hd[1],l\_hd[0]))

fe["loser\_hd"]=fe["loser\_hand"]

fe["loser\_hd"].replace(l\_huid,inplace=True)

tour\_level=pd.unique(fe[['tourney\_level']].values.ravel('K'))

tour\_l=pd.factorize(tour\_level)

tour\_uid=dict(zip(tour\_l[1],tour\_l[0]))

fe["tour\_uid"]=fe["tourney\_level"]

fe["tour\_uid"].replace(tour\_uid,inplace=True)

X = np.array(fe.drop(['winner\_name','loser\_name','surface','winner','winner\_hand','loser\_hand','tourney\_level'], 1))

y = np.array(fe['winner'])

X = preprocessing.scale(X)

X\_train, X\_test, y\_train, y\_test = cross\_validation.train\_test\_split(X, y, test\_size=0.2)

#LinearRegression

clf = LinearRegression()

clf = LinearRegression(n\_jobs=-1)

clf.fit(X\_train, y\_train)

confidence = clf.score(X\_test, y\_test)

print(confidence)

clf.predict(X\_test)

#Output:1.0

#KNearestNeighborsClassifier

knn=neighbors.KNeighborsClassifier()

#knn = neighbors.KNeighborsClassifier(n\_jobs=-1)

knn.fit(X\_train, y\_train)

accuracy = knn.score(X\_test, y\_test)

print(accuracy)

knn.predict(X\_test)

#Output:0.13

#Decision Tree

dc=tree.DecisionTreeClassifier()

dc.fit(X\_train, y\_train)

confidence = dc.score(X\_test, y\_test)

print(confidence)

dc.predict(X\_test)

dt\_train\_gini=tree.DecisionTreeClassifier(criterion="gini",random\_state=100,max\_depth=4,min\_samples\_leaf=5)

dt\_train\_gini.fit(X\_train,y\_train)

#Output:0.83

from sklearn import tree

with open("dt\_train\_gini.txt","w") as f:

f=tree.export\_graphviz(dt\_train\_gini,out\_file=f)

#NaiveBayes Classifier

from sklearn.naive\_bayes import GaussianNB

model=GaussianNB()

model.fit(X,y)

confidence = dc.score(X\_test, y\_test)

print(confidence)

predicted=model.predict(X\_test)

predicted

#Output:0.82